

Capitolo 1

Introduzione al Software Engineering Body of Knowledge (SWEBOK)¹

Ercole Colonese, Consulente
2004

A dispetto dei milioni di professionisti del software che operano a livello mondiale e della vastità del software presente nella società moderna, l'ingegneria del software non ha raggiunto ancora lo stato di legittima disciplina e di professione riconosciuta.

Solo dal 1993 un'apposita organizzazione, il Software Engineering Coordinating Committee (SWECC) opera attivamente per promuovere la professione del *Software Engineer* (Ingegnere del software) e della disciplina del *Software Engineering* (Ingegneria del software). Il comitato è stato fondato dall'IEEE Computer Society e dall'Association for Computing Machinery (ACM).

Un primo traguardo nel definire una disciplina è quello di raggiungere un consenso sui contenuti di base della professione (core body of knowledge). Operando in questa direzione, il comitato ha definito il corpo delle competenze di base della nuova professione.

Questo documento fornisce una descrizione di massima di tali competenze. Ciascuna competenza sarà descritta in successivi documenti in fase di preparazione.

Software Engineering nelle organizzazioni software

Questo documento introduttivo, insieme a tutti quelli che lo seguiranno sulle singole competenze, ha lo scopo di diffondere all'interno della comunità del software i concetti di base dell'ingegneria del software e di contribuire alla costituzione di una base di conoscenza comune sull'argomento. In ogni documento un capitolo apposito, "Software Engineering nelle organizzazioni software", fornirà gli elementi particolari (esempi, linee guida, ecc.) per calare nella realtà quotidiana i concetti di base trattati. In molti casi la soluzione adottata nella pratica quotidiana sarà molto più semplice di quanto la teoria descriva. E' importante tuttavia che essa sia conosciuta in tutta la sua completezza (e, a volte complessità) perché si capisca la portata e la validità della soluzione proposta.

¹ Quanto descritto in questo documento è tratto da "Guide to the Software Engineering Body of Knowledge, SWEBOK, A Project of the Software Engineering Coordinating Committee. IEEE-Trial Version 1.00–May 2001".

Nota: SWEBOK è un marchio ufficiale del servizio IEEE.

Che cos'è la Software Engineering?

La IEEE Computer Society definisce l'ingegneria del software come:

“(1) L'applicazione di un sistematico, disciplinato e quantificabile approccio allo sviluppo, all'operatività e alla manutenzione del software; cioè l'applicazione dell'ingegneria al software.

(2) Lo studio degli approcci definiti al punto precedente.”²

Che cos'è una professione riconosciuta?

Affinché l'ingegneria del software sia legittimata come disciplina dell'ingegneria e sia riconosciuta come professione occorre raggiungere il consenso sulle competenze di base (core body of knowledge).

Quali sono le caratteristiche di una professione?

Le caratteristiche principali della professione ingegneristica³ includono le seguenti componenti:

- *Formazione iniziale* eseguita all'interno di un curriculum validato da un'organizzazione apposita tramite l'*accreditazione* (es.: frequenza dei corsi di un piano di formazione definito e superamento degli esami previsti);
- Registrazione dell'aderenza alle pratiche riconosciute tramite il superamento di esami particolari (es.: iscrizione all'albo degli ingegneri, certificazione interna ecc.);
- Sviluppo continuo delle competenze tramite frequenza di corsi di specializzazione;
- Supporto da parte di una comunità accreditata;
- Impegno a seguire un comportamento etico.

L'insieme degli articoli che ci si appresta a scrivere e a divulgare intendono coprire i primi tre punti.

Quali obiettivi si pone il progetto SWEBOK?

Questo articolo ed i successivi non intendono sostituirsi alla conoscenza stessa. Essa esiste già nella vasta letteratura disponibile. S'intende invece selezionare e proporre quanto di più consolidato ed efficace esista e presentarlo in maniera organizzata.

Organizzazione del materiale

Il materiale in via di produzione è organizzato secondo le dieci competenze riconosciute dall'ingegneria del software:

1. Software requirements
2. Software design
3. Software construction
4. Software testing
5. Software maintenance

² "IEEE Standard Glossary of Software Engineering Terminology, 1990."

³ G. Ford and E. Gibbs, "A Mature Profession of Software Engineering" Software Engineering Institute, Carnegie Mellon University, Pittsburg, Pennsylvania, January 1996.

6. Software configuration management
7. Software engineering management
8. Software engineering process
9. Software engineering tools and methods
10. Software quality

Per fare maggiore chiarezza e sgomberare il campo da equivoci molto comuni, diciamo che la software engineering opera in maniera sinergica (e spesso ne condivide i confini) con altre scienze adiacenti (ma è cosa distinta da esse) come:

Cognitive sciences and human factors
Computer engineering
Computer science
Management science
Mathematics
Project management
Systems engineering

Riferimenti

Per ciascuna competenza discussa saranno forniti riferimenti alla letteratura disponibile. Sia per consentire approfondimenti a chi ne senta l'esigenza sia per fornire riferimenti autorevoli a proposito della materia trattata. La scelta e l'estensione dei riferimenti sono soggettive e non pretendono di essere esaustive.

Livello di approfondimento

Il livello di approfondimento cui arrivare con la presentazione di un argomento è questione antica e di difficile risoluzione. Una definizione accettata dall'ingegneria del software è: *“Il livello di conoscenza generalmente accettato è quello che permette di gestire con successo la maggior parte dei progetti nella maggior parte delle situazioni reali”*.

Noi cercheremo di scendere ad un livello di approfondimento tale che permetta alle persone di apprendere gli elementi basilari di ciascun argomento in modo da poter svolgere con profitto (e successo) le proprie attività quotidiane. Lasciando a ciascuno il compito di approfondire gli argomenti secondo le proprie esigenze.

Ratings

L'ingegneria del software classifica i vari argomenti trattati secondo i livelli e le categorie pedagogiche del modello Benjamin Bloom⁴. Il concetto che sta dietro questa metodologia è che il processo di apprendimento si sviluppa secondo livelli successivi di approfondimento: conoscenza, comprensione, applicazione, analisi, sintesi e valutazione. Ciascun argomento trattato, quindi, dovrebbe essere associato ad un livello. Ad esempio, per quanto riguarda la competenza relativa ai requisiti

software, l'argomento specifico “Processo dei requisiti software” è associato al primo livello: “conoscenza”. L'argomento “Origine dei requisiti” è associato al livello 2 “comprensione”, mentre l'argomento “Tecniche di analisi dei requisiti” è associato al livello 3 “applicazione”.

Questo argomento, discusso qui solo per amore di completezza e per fornire uno spunto di approfondimento a chi è particolarmente interessato alle tematiche dell'apprendimento, non sarà trattato negli articoli che seguiranno. Sarà argomento di analisi successive per verificarne il reale beneficio.

AREE DI COMPETENZA (KNOWLEDGE AREAS)

Di seguito è fornita una breve descrizione di ciascuna delle dieci aree di competenza. Le prime cinque sono riportate nell'ordine in cui generalmente sono svolte nel ciclo di sviluppo (a cascata). Le rimanenti cinque sono generalmente svolte lungo l'intero ciclo di sviluppo e sono riportate in ordine alfabetico. Ciascuna area di competenza è composta da più sotto-aree che indirizzano argomenti specifici.

Software Requirements

Un requisito è definito come la proprietà da esibire per dimostrare che si risolve un problema nel mondo reale. Sono definite sei sotto-aree di competenza:

- Requirements engineering process
- Requirements elicitation
- Requirements analysis
- Software Requirements specifications
- Requirements validation
- Requirements management

La prima sotto-area di conoscenza “*Requirements engineering process*” stabilisce la necessità di definire un processo per i requisiti ed è propedeutica alle altre cinque sotto-aree. La sotto-area descrive il modello di processo da adottare, gli attori da coinvolgere, il supporto richiesto, il tipo di management necessario e il miglioramento della qualità del processo stesso.

La seconda sotto-area “*Requirements elicitation*” tratta l'identificazione dell'origine dei requisiti e come essi possono essere collezionati. Include le tecniche da adottare per una corretta esecuzione della fase di “*elicitation*”.

La terza sotto-area “*Requirements analysis*” tratta delle tecniche per l'analisi dei requisiti allo scopo di:

- rilevare e risolvere eventuali conflitti tra i requisiti;
- definire i confini del sistema e come esso interagisce con l'ambiente circostante;
- ricavare i requisiti del software elaborando i requisiti del sistema.

Questa sotto-area include le attività relative alla classificazione dei requisiti, modellazione concettuale, progettazione dell'architettura, allocazione dei requisiti e negoziazione dei requisiti.

⁴ Bloom, B.S. (Ed.) 1956, *Taxonomy of educational objectives: The classification of educational goals: Handbook I, cognitive domain*. New York; Toronto: Longmans, Green.

La quarta sotto-area “*Software requirements specification*” descrive la struttura, qualità e verificabilità del documento dei requisiti. Il documento deve contenere due parti distinte rivolte a due lettori differenti e con due obiettivi diversi: “definizione dei requisiti del sistema” e “specificazione dei requisiti del software”. La sotto-area descrive anche la struttura dei documenti in oggetto, lo standard relativo e la qualità.

La quinta sotto-area “*Requirements validation*” ha il compito di scoprire i possibili problemi legati ai requisiti prima che si assumano impegni su di essi. La validazione consiste nella revisione della documentazione dei requisiti per verificare se il sistema definito sia corretto (se i requisiti espressi dal cliente siano correttamente indirizzati). La validazione indirizza quindi la revisione dei documenti dei requisiti, dei prototipi, la validazione del modello ed i test di accettazione.

La sesta ed ultima sotto-area “*Requirements management*” spazia sull’intero ciclo di vita del software. Tratta principalmente della gestione delle modifiche ai requisiti e fotografa accuratamente lo stato attuale del software. Include la gestione delle modifiche, gli attributi dei requisiti e la registrazione dei requisiti.

Software Design

La progettazione del software spazia sull’intero ciclo di vita del software. Fondamentalmente tratta della gestione delle modifiche ai requisiti in ottica di come il software è, è stato o sarà costruito. Sono definite sei sotto-aree di competenza:

- Basic concepts
- Key issues of software design
- Structure and architecture
- Software design quality analysis and evaluation
- Design notation
- Software design strategies and methods

La prima sotto-area “*Basic concepts*” definisce i concetti basilari e le nozioni relative alla progettazione del software. Si tratta di concetti generali sull’ambito di applicazione della progettazione del software, il processo di progettazione e le relative tecniche.

La seconda sotto-area “*Key issues of software design*” indirizza la concorrenzialità, il controllo e la gestione degli eventi, la gestione degli errori e delle situazioni eccezionali e loro distribuzione, l’interattività dei sistemi e loro persistenza.

La terza sotto-area “*Structure and architecture*” indirizza le problematiche relative alla struttura ed architettura del software, ai diversi punti di vista da cui essa può essere valutata, agli stili e ai disegni (*pattern*) architetturali, alle famiglie di programmi, ambienti e strutture architetturali (*design frameworks*).

La quarta sotto-area “*Software design quality analysis and evaluation*” indirizza gli elementi principali della qualità di una progettazione: attributi della qualità, analisi della qualità, misure e strumenti per la valutazione. Il

quadro generale della qualità del software è descritta nell’area specifica “*Software quality*”.

La quinta sotto-area “*Design notations*” indirizza le notazioni utilizzate nella progettazione del software divise in due categorie: descrizioni strutturali e di comportamento.

La sesta ed ultima sotto-area indirizza “*Software design strategies and methods*”. In primo luogo sono descritte le strategie generali, quindi i metodi function-oriented, poi quelli object-oriented ed infine i metodi orientati alla data-structure e quelli transformational.

Software Construction

La realizzazione del software è una parte fondamentale dell’ingegneria del software: stesura del codice e realizzazione delle strutture dati, produzione della documentazione tecnica, verifica (*reviews*) e validazione (*unit testing*). Il primo e più importante metodo da adottare per scomporre la realizzazione del software in unità più semplici da gestire è quello di riconoscere l’esistenza dei quattro principi generali che maggiormente influenzano il modo in cui il software è realizzato.

Questi principi sono: Riduzione della complessità, Anticipazione della diversità, Strutturazione per la validazione, Utilizzo di standard esterni.

Un secondo ma non meno importante metodo di scomporre il problema è quello di riconoscere l’esistenza di tre metodi/stili per la costruzione del software: Linguistico, Formale e Visuale.

L’area di competenza fornisce una descrizione dei due punti di vista per la scomposizione del problema. In particolare, sono definite quattro sotto-aree di competenza:

- Reduction in complexity
- Anticipation of diversity
- Structuring for validation
- Use of external standards

Software Testing

Il test del software consiste nella verifica dinamica del comportamento del software realizzato tramite un numero finito di casi di prova selezionati tra un numero pressoché infinito di modalità diverse di utilizzo e a fronte di un comportamento specificato ed atteso. Sono definite cinque sotto-aree di competenza:

- Basic concepts
- Test levels
- Test techniques
- Test-related measures
- Managing the test process

La prima sotto-area “*Basic concepts*” presenta la terminologia del testing, i fondamenti teorici del testing e le relazioni con le altre attività del ciclo di vita del software.

La seconda sotto-area “*Test levels*” indirizza i target e gli obiettivi dei diversi test.

La terza sotto-area “*Test techniques*” descrive due categorie di tecniche: la prima categoria raggruppa le tecniche basate sui criteri con cui i test sono generati, la seconda categoria raggruppa le tecniche normalmente ignorate in fase di implementazione. Sono forniti suggerimenti su come selezionare e combinare le diverse tecniche in base ai target ed agli obiettivi da raggiungere.

La quarta sotto-area “*Test-related measures*” divide le misure tra quelle per valutare il software testato e quelle per valutare l’efficacia dei test eseguiti.

La quinta ed ultima sotto-area “*Managing the test process*” descrive le attività principali e gli elementi da prendere in considerazione.

Software Maintenance

La manutenzione del software inizia appena esso è rilasciato in esercizio ed indirizza la risoluzione dei problemi rilevati dagli utenti, le modifiche dell’ambiente operativo, i nuovi requisiti. Sono definite quattro sotto-aree di competenza:

- Basic concepts
- Maintenance process
- Key issues
- Techniques for maintenance

La prima sotto-area “*Basic concepts*” descrive i concetti di base, le definizioni, le attività principali ed i relativi problemi legati alla manutenzione del software.

La seconda sotto-area “*Maintenance process*” descrive il processo tipico di manutenzione del software basandosi sullo standard IEEE 1219 e ISO/IEC 14764.

La terza sotto-area “*Key issues*” indirizza le problematiche tipiche della manutenzione del software. Gli aspetti indirizzati sono quelli tecnici, gestionali, relativi a stime e costi, misure.

Le quarta ed ultima sotto-area “*Techniques for maintenance*” indirizza le tecniche relative a: comprensione del software, reingegnerizzazione del software, reverse engineering e analisi degli impatti.

Software Configuration Management

Il Software Configuration Management (SCM) è la disciplina atta a identificare la configurazione software di un sistema in diversi momenti della sua vita per consentire il controllo sistematico delle modifiche e mantenere l’integrità e la tracciabilità della configurazione stessa. Sono definite sei sotto-aree di competenza:

- Management of the SCM process
- Software configuration identification
- Software configuration control
- Software configuration status accounting
- Software configuration auditing
- Software release management and delivery

La prima sotto-area “*Management of the SCM process*” indirizza il contesto organizzativo del SCM, fornisce linee guida per il SCM, definisce le modalità per la pianificazione SCM ed il piano SCM in particolare, per la sorveglianza dell’implementazione del SCM.

La seconda sotto-area “*Software configuration identification*” descrive come identificare gli elementi da controllare, stabilisce gli schemi per l’identificazione degli elementi e le loro versioni, stabilisce le tecniche e gli strumenti da adottare per l’acquisizione ed il controllo degli elementi identificati.

La terza sotto-area “*Software configuration control*” indirizza la gestione delle modifiche durante il ciclo di vita del software. Le attività si articolano su tre fasi successive: richiesta, valutazione e approvazione delle modifiche al software; implementazione delle modifiche; deviazioni e rinunce.

La quarta sotto-area “*Software configuration status accounting*” descrive le modalità per la rilevazione delle informazioni relative allo stato della configurazione e la produzione della relativa reportistica.

La quinta sotto-area “*Software configuration auditing*” descrive le attività di verifica della configurazione funzionale e fisica del software, della baseline in corso.

La sesta ed ultima sotto-area “*Software release management and delivery*” copre le attività relative alla costruzione del software ed alla gestione del suo rilascio in esercizio.

Software Engineering Management

La gestione dell’ingegnerizzazione del software è assimilabile, da un lato, alla gestione di qualsiasi altro processo complesso. D’altro lato, esistono molti aspetti particolari e specifici che rendono la gestione della produzione del software e dei suoi processi sicuramente più complicata. Sono definite tre sotto-aree di competenza:

- Organizational management
- Process/Project management
- Software engineering measurement

La prima sotto-area “*Organizational management*” comprende la gestione delle politiche (*policy*), del personale, delle comunicazioni, del portafoglio software, dei fornitori.

La seconda sotto-area “*Process/Project management*” indirizza le fasi tipiche di inializzazione e definizione dell’ambito, pianificazione, partenza (*kick-off*), revisione e valutazione, chiusura.

La terza ed ultima sotto-area “*Software engineering measurement*” copre i principi generali delle misure del software. Sono indirizzati, nell’ordine, gli elementi relativi a: definizione degli obiettivi del programma di misure; selezione delle misure utili; misurazione del software e collezione dei dati; modelli di metriche del software.

Software Engineering Process

L'area di competenza indirizza la definizione, implementazione, misurazione, gestione, modifica e miglioramento del processo di ingegneria del software. Sono definite sei sotto-aree di competenza:

- Basic concepts
- Process infrastructure
- Measurements specific to software engineering process
- Process definition
- Qualitative process analysis
- Process implementation and change

La prima sotto-area "*Basic concepts*" presenta i concetti di base, i temi e la terminologia.

La seconda sotto-area "*Process infrastructure*" descrive le responsabilità del gruppo preposto alla gestione del processo in oggetto "*Software Engineering Process Group (SEPG)*" e l'organizzazione di sviluppo del software (*Software Factory*).

La terza sotto-area "*Measurements specific to software engineering process*" presenta la metodologia ed i paradigmi legati alle misurazioni fatte sul campo.

La quarta sotto-area "*Process definition*" descrive le conoscenze relative a tale pratica: i vari tipi di definizione del processo, i modelli del ciclo di vita del software (*life cycle*), le notazioni usate per rappresentare tali definizioni, i metodi per la definizione dei processi e l'automazione relativa alle varie definizioni.

La quinta sotto-area "*Qualitative process analysis*" descrive la revisione della definizione dei processi e l'analisi delle cause di eventuali non conformità o necessità di modifica (*root cause analysis*).

La sesta ed ultima sotto-area "*Process implementation and change*" descrive i paradigmi e fornisce le linee guida per l'implementazione dei processi, la gestione delle loro modifiche e la valutazione dei risultati.

Software Engineering Tools and Methods

L'area di competenza indirizza gli strumenti a supporto dello sviluppo e manutenzione del software, ed i metodi per lo sviluppo del software. Gli argomenti sono trattati secondo le dieci aree di competenza definite (Requirements, Design, Construction, Testing, Maintenance, Process, Quality, Configuration management, Management).

Sono definite due sezioni:

- Software development environments
- Software development methods

La sezione "*Software development environments*" comprende l'insieme degli strumenti (tool) a supporto del processo di sviluppo software. La presentazione dei "Software tools" è fatta secondo le già citate aree di competenza definite. A queste si aggiungono altre due categorie di tool: strumenti per il supporto infrastrutturale, che non corrisponde ad alcuna delle aree

di competenza sopra elencate; un insieme di strumenti e tecniche per l'integrazione del software, che è potenzialmente applicabile a tutte le aree precedenti. La sezione "*Software development methods*" indirizza quattro categorie di metodi: metodi euristici relativi ad approcci informali, metodi formali relativi ad approcci matematici, metodi prototipali relativi a tecniche di sviluppo software basate su approcci prototipali, miscellanea di metodi discussi nelle loro generalità.

Software Quality

Questa area di competenza descrive le considerazioni generali sulla qualità del software che prescindono dal processo di sviluppo adottato. Poiché la qualità del software ha aspetti diversi, essa è trattata in molte aree di competenza. In quest'area si fa riferimento a tali aspetti ed alle relative aree di competenza in cui sono approfonditi i vari temi.

Sono definite quattro sotto-aree di competenza:

- Software quality concepts
- Purpose and planning of SQA and V&V
- Activities and techniques for SQA and V&V
- Measurement applied to SQA and V&V

La prima sotto-area "*Software quality concepts*" descrive i concetti base della qualità del software, le metriche, il valore della qualità, le norme ISO 9126, le dipendenze ed altre necessità specifiche sul tema.

La seconda sotto-area "*Purpose and planning of SQA (Software Quality Assurance) and V&V (Verification and Validation)*" descrive le attività in oggetto, la loro pianificazione ed i piani specifici di SQA e di V&V.

La terza sotto-area "*Activities and techniques for SQA and V&V*" descrive le attività e le tecniche relative all'assicurazione della qualità del software ed alla verifica e validazione. La descrizione include le tecniche statiche (revisioni), dinamiche (testing) ed altre tecniche per la SQA e test di V&V.

La quarta ed ultima sotto-area "*Measurement applied to SQA and V&V*" descrive i fondamentali delle misure, le misurazioni stesse, le tecniche di analisi delle misure, la caratterizzazione dei difetti, altri usi dei dati relativi alle misure SQA e V&V.

BIBLIOGRAFIA

Di seguito è riportata la bibliografia essenziale sui temi generali del Software Engineering. Ulteriori riferimenti saranno forniti nei successivi capitoli dedicati ai singoli argomenti trattati.

[1] Guide to the Software Engineering Body of Knowledge, SWEBOK, A Project of the Software Engineering Coordinating Committee. IEEE-Trial Version 1.00 (May 2001).

[2] McConnell Steve – Professional Software Development – Addison-Wesley (2003).

[3] Capability Maturity Model Integration (CMMI), Version 1.1, Staged representation – March 2002 –

Capitolo 1 – Introduzione al Software Engineering Body of Knowledge (SWEBOK)

Software Engineering Institute - Carnegie Mellon University.

- [4] Pressman Roger S. - Software Engineering, A Practitioner's Approach - McGraw-Hills (2000).
- [5] James Martin - Information Engineering Books: I (introduction), II (planning and analysis), III (design and construction) - Prentice Hall (1990).
- [6] Brooks Frederick .P., Jr - The mythical Man-Month: Essays on Software Engineering – Anniversary Edition Addison-Wesley (1995).
- [7] ISO/IEC 9126:2001, Information technology – Software product evaluation – Quality characteristics and guidelines for their use.
- [8] UNI EN ISO 9001:2000, Sistema qualità – Modello per l'assicurazione della qualità nella progettazione, sviluppo, fabbricazione, installazione e assistenza – Guida all'uso.
- [9] ISO 9000-3:2004, Linee guida per l'applicazione di ISO 9001 allo sviluppo, fornitura e manutenzione del software.
- [10] ISO 14598: Valutazione del prodotto software.
- [11] ISO 12207:1995, Ciclo di vita del software.
- [12] IEEE Standard Glossary of Software Engineering Terminology, 1990.
- [13] G. Ford and E. Gibbs, A Mature Profession of Software Engineering, Software Engineering Institute, Carnegie Mellon University, Pittsburg, Pennsylvania, January 1996.

REVISIONI

Il documento è stato sottoposto a revisione interna non formale.